AzureOP:基于 WindowsAzure 云并行计算的期权定价 SaaS *

林溢星 [1,2]，赵 地 [3†]，迟学斌 [1,2,4]，姜金荣 [1,4]

(1. 中国科学院计算机网络信息中心 高性能计算部，北京 100190; 2. 中国科学院大学，北京 100049; 3. 中国科学院计算技术研究所，北京 100190; 4. 中国科学院计算科学应用研究中心，北京 100190)

摘 要： 计算速度对于期权交易者至关重要，关系到如何有效地制定价格并评估相应的风险，而云并行计算提供的随收随付制（pay-as-you-go）可以实现低成本运行。在微软云平台 Windows Azure 的基础上，开发了基于云并行计算的期权定价试点云软件 AzureOP，该软件以较低的费用提供了低风险和高速度，并给出了 AzureOP 对于美式期权价格的模拟结果，绘制了对应的期权价格定价曲线和定价曲面。最后，对云并行计算在金融应用上的优势和不足进行了总结和讨论，同时举例说明了试点云软件 AzureOP 的具体细节。

关键词： 云计算；并行计算；SaaS 开发；计算金融；期权定价

中图分类号： TP301.4　　doi: 10.3969/j.issn.1001-3695.2018.01.0016

AzureOP: parallel computing based SaaS for option pricing on cloud

Lin Yixing[1, 2], Zhao Di[3†], Chi Xuebin[1, 2, 4], Jiang Jinrong[1, 4]

(1. *Computer Network Information Center*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*; 2. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*; 3. *Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*; 4. *Center of Scientific Computing Applications & Research*, *Chinese Academy of Sciences*, *Beijing* 100190, *China*)

Abstract: The computational speed is important for the crucial for option traders to efficiently decide the price and evaluate the corresponding risk, and cloud parallel computing provides a low-cost implementation by providers' Pay-as-You-Go policy. Based on Microsoft's cloud platform Windows Azure, developed a option pricing's pilot cloud software - AzureOP, this software provided low risk and high speed with a lower cost, and presented simulation results of AzureOP for American Put, plotted the corresponding option pricing curve and pricing surface. Finally, makes conclusions and discusses the advantage and the disadvantage of computational finance on the cloud, and illustrates the details development of pilot cloud software AzureOP.

Key words: Cloud computing; parallel computing; SaaS development; computational finance; option pricing

0 引言

虽然云计算被定义为计算和存储交付，但作为基于互联网的计算[1-4]，云计算也可以被定义为金融公司和制造公司的基于计算的服务器：计算密集的部分在服务器（云）上完成，结果通过 Web 接口交付给用户。众所周知，云计算包含三种服务模式：基础架构即服务（IaaS），平台即服务（PaaS）和软件即服务（SaaS）。当前的公开 PaaS 包括 Microsoft Windows Azure，Amazon Web Services，Google App Engine 和 Sun Grid Compute Utility [5]以及 Open Cirrus [6]等新兴实验平台。PaaS 的一个例子如图 1 所示。此 PaaS 基于微软的数据中心（IaaS），逻辑结构由 Windows Azure 的 HPC 调度程序和 Azure 节点组成。选择好 PaaS 提供商后，SaaS 得到开发，测试和交付，并且成功发布的 SaaS 包括生物信息学[7-16]，图像处理[17]，工程[18]，医学信息[19]，娱乐[20]，金融，社会科学等。

计算金融是一个跨学科的领域，将数理统计建模、分析和计算应用于定价、交易和对冲决策，并分析这些决策的风险。这个领域典型的数理统计技术包括随机分析，蒙特卡洛模拟，微分方程，多元分析，时间序列等。很多通用软件兼容了计算金融，例如 MATLAB，Mathematica，Maple 和 R，另外还有专

收稿日期： 2018-01-04；**修回日期：** 2018-03-09　**基金项目：** 国家重点研究发展计划资助项目（2016YFB0201400，2016YFB0200800）；国家自然科学基金重大研究计划资助项目（91530324）；北京市自然科学基金资助项目（4161004）；北京市科技资助项目（Z161100000216143，Z171100000117001）；中国科学院知识创新项目（XXH13504-03-02）

作者简介： 林溢星（1992-），女，福建莆田人，硕士，主要研究方向为脑科学、深度学习；赵地（1978-），男（通信作者），副研究员，博士，主要研究方向为类脑计算、深度学习（zhaodi@escience.cn）；迟学斌（1963-），男，研究员，博士，主要研究方向为高性能计算；姜金荣（1977-），男，研究员，博士，主要研究方向为并行计算.

门用于计算金融的软件，例如 Quantifi，StreamBase，Volera 和 SciFinance。

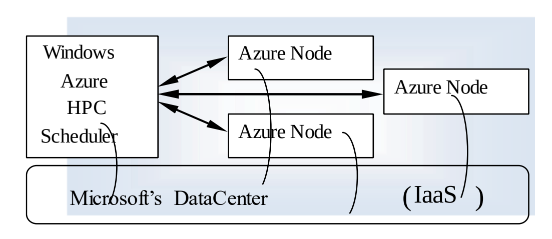


图 1　Windows Azure HPC 的逻辑结构

除台式电脑外，并行计算还被广泛应用于计算金融领域，例如集群和 GPU 等平台。云并行计算是云上的并行计算模式，如 Windows Azure 的 HPC 调度程序，它为金融公司和专业人员提供了可扩展和低成本的并行计算。

期权是一种能在未来某特定时间以特定价格买入或卖出一定数量的某种特定商品的权利。期权可以通过 Black-Scholes 方程或随机波动模型进行定价，并通过二项式树、蒙特卡洛模拟和有限差分法等数值方法求解。期权定价的计算速度是交易双方取得成功的关键因素之一，云并行计算为满足期权定价的速度需求提供了新的平台。本文在 Windows Azure 的基础上，利用 Black-Scholes 方程模拟期权定价，并且开发了基于有限差分软件的云并行计算 AzureOP 进行期权定价。

1　AzureOP 方法

首先介绍如何在云上使用 AzureOP 进行期权定价。然后介绍如何设计 AzureOP：包括三种 Black-Scholes 方程的数值求解方案，这三种方案的云并行实现，以及云上使用到的软件，并且在第一次运行之前，将软件部署到 Windows Azure。最后介绍 AzureOP 的计算结果。

1.1　软件

由于 AzureOP 是基于云平台 Azure 的软件，安装过程由平台配置过程组成，具体细节将在后面讨论。在成功部署之后，除了超级计算和背后的大数据存储的功能，AzureOP 的使用类似于任何桌面软件。

AzureOP 的 GUI 如图 2 所示，AzureOP 的界面由三部分组成："选择选项"，"参数设置"和命令按钮。选项样式可以从"option selection"部分中选择，参数可以从"parameter setting"部分进行设置。之后，按下"start"按钮启动 AzureOP。 AzureOP 的中间结果存储在 Azure blob 中，按下"Fetch Results"按钮可以将计算结果提取到 AzureOP 的输出帧。

1.2　模型

AzureOP 通过求解 Black-Scholes 方程来做期权定价。Black-Scholes 模型是一个二阶偏微分方程，它描述了期权定价从终止条件递归到价格为零的结束时期的过程。从原始文献[21]推导出 Black-Scholes 方程后，得到：

$$\frac{\partial V}{\partial t}+\frac{1}{2}\delta^2 S^2\frac{\partial^2 V}{\partial S^2}+rS\frac{\partial V}{\partial S}-rV=0, \tag{1}$$

其中：V 是期权价格，S 是股票价格，t 是时间，δ 是股票收益率的波动率，r 是利率。方程（1）可以进一步简化为更简单的形式，如热传导方程。美式期权的左边界条件为

$$V_0^t=K, \tag{2.1}$$

而欧氏期权则为

$$V_0^t=Ke^{-(T-t)}, \tag{2.2}$$

其中 K 是常数，看涨期权的左边界条件为

$$V_0^t=0, \tag{3.1}$$

并且期权的右边界条件为

$$V_S^t=0, \tag{3.2}$$

期权价格预测的最终条件为

$$V_S^T=max(K-S,0), \tag{4}$$

其中 T 是时间的步长总数。Black-Scholes 方程的关键思想是一方可以用方程式中的"正确"价格来对冲资产以避免所有的风险，所以定价的计算速度很重要。Black-Scholes 方程的求解有两种方法：分析方法和数值方法。文献[21]中，Martin 等人开发了近似解析解，而 Black-Scholes 方程的数值解法包括有限差分法[22,23]、蒙特卡洛模拟[24~26]、路径积分[27]、机器学习[28]等。本文用有限差分法求解 Black-Scholes 方程。

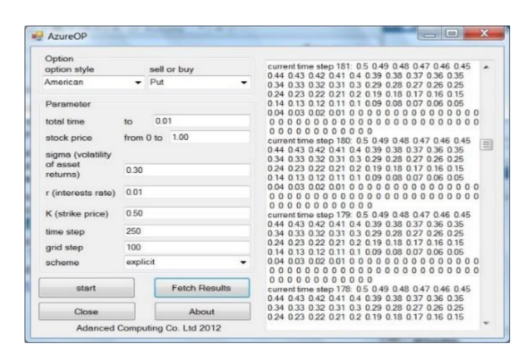


图 2　AzureOP 的界面

1.3　显式方案

AzureOP 由三个方案组成，用于数值求解 Black-Scholes 方程：显式方案、隐式方案和 Crank-Nicolson 方案。通过显式方案离散 Black-Scholes 方程，本文定义下面的算子：

$$\frac{\partial V}{\partial t}=\frac{V_i^{n+1}-V_i^n}{\Delta t}, \tag{5.1}$$

$$\frac{\partial^2 V}{\partial S^2}=\frac{V_{i+1}^{n+1}-2V_i^{n+1}+V_{i-1}^{n+1}}{(\Delta S)^2}, \tag{5.2}$$

$$\frac{\partial V}{\partial S}=\frac{V_{i+1}^{n+1}-V_{i-1}^{n+1}}{2\Delta S}, \tag{5.3}$$

其中：n 是时间步长的计数器，i 是网格（股票价格）的计数器。将这些算子（5.1-5.3）代入方程（1）得到

$$V_i^n=A_iV_{i-1}^{n+1}+B_iV_i^{n+1}+C_iV_{i+1}^{n+1}, \tag{6.1}$$

其中，

$$A_i=\frac{1}{1+r\Delta t}\left(\frac{\delta^2S^2\Delta t}{2(\Delta S)^2}-\frac{rS\Delta t}{2\Delta S}\right), \tag{6.2}$$

$$B_i=\frac{1}{1+r\Delta t}\left(1-\frac{\delta^2S^2\Delta t}{(\Delta S)^2}\right), \tag{6.3}$$

$$C_i = \frac{1}{1+r\Delta t}\left(\frac{\delta^2 S^2 \Delta t}{2(\Delta S)^2} + \frac{rS\Delta t}{2\Delta S}\right) \tag{6.4}$$

由方程（6.1）可以得到，期权价格从最终状态（4）向当前时间步长 n 后向递归演变，每个网格（股票价格）的期权价格从 1 到 I-1 进行逐一计算，其中 I 是网格步长的总数。期权价格预测的左边界条件可以被离散化为：

$$V_0^n = Ke^{-(T-n\Delta t)}, \tag{7.1a}$$

或者

$$V_0^n = 0, \tag{7.1b}$$

期权价格方程（3.2）的右边界条件可以被离散化为：

$$V_I^n = 0 \tag{7.2}$$

其中V_0^n和V_I^n不需要计算，可以直接用边界条件代替。另外，为了保持显式方案的稳定性，应该满足条件$\Delta t \le \frac{(\Delta S)^2}{2}$。

1.4　隐式方案

AzureOP 还可以通过隐式方案离散 Black-Scholes 方程，算子定义如下：

$$\frac{\partial^2 V}{\partial S^2} = \frac{V_{i+1}^n - 2V_i^n + V_{i-1}^n}{(\Delta S)^2}, \tag{8.1}$$

$$\frac{\partial V}{\partial S} = \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta S}, \tag{8.2}$$

将方程（5.1），（8.1）和（8.2）代入 Black-Scholes 方程（1），可以得到：

$$V_i^{n+1} = A_i V_{i-1}^n + B_i V_i^n + C_i V_{i+1}^n \tag{9.1}$$

其中：

$$A_i = \frac{\Delta t}{2}\left(-\frac{\delta^2 S^2}{(\Delta S)^2} + \frac{rS}{\Delta S}\right), \tag{9.2}$$

$$B_i = 1 + \frac{\delta^2 S^2 \Delta t}{(\Delta S)^2} + r\Delta t, \tag{9.3}$$

$$C_i = -\frac{\Delta t}{2}\left(\frac{\delta^2 S^2}{(\Delta S)^2} + \frac{rS}{\Delta S}\right). \tag{9.4}$$

将方程（9.1）转换成矩阵形式，可以得到

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{I-2} & B_{I-2} & C_{I-2} \\ 0 & 0 & 0 & \cdots & 0 & A_{I-1} & B_{I-1} \end{bmatrix} \begin{bmatrix} V_1^n \\ V_2^n \\ \vdots \\ V_{I-2}^n \\ V_{I-1}^n \end{bmatrix} = \begin{bmatrix} V_1^{n+1} - A_1 V_0^n \\ V_2^{n+1} \\ \vdots \\ V_{I-2}^{n+1} \\ V_{I-1}^{n+1} - C_{I-1} V_I^n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{I-2} \\ F_{I-1} \end{bmatrix} \tag{10}$$

其中 F_i 是线性系统方程（10）的等式右侧部分。为了求解系统方程（10），使用三对角矩阵算法（TDMA）。与显式方案相比，隐式方案在任何 Δt 和 Δs 情况下都更稳定。

另外，由于线性系统方程（10）的系数矩阵有三对角线（左对角线，中对角线和右对角线），所以这些对角线的信息被压缩成三个矢量（左矢量，中矢量和右矢量），并且忽略所有的零。因此，在一个 I*I 的系数矩阵中，左矢量的大小为 I-1，中矢量的大小为 I，右矢量的大小为 I-1。

1.5　Crank-Nicolson 方案

AzureOP 求解 Black-Scholes 方程的第三种内置方案是 Crank-Nicolson 方案，算子定义如下：

$$\frac{\partial^2 V}{\partial S^2} = \frac{(V_{i+1}^{n+1} - 2V_i^{n+1} + V_{i-1}^{n+1}) + (V_{i+1}^n - 2V_i^n + V_{i-1}^n)}{2(\Delta S)^2}, \tag{11.1}$$

$$\frac{\partial V}{\partial S} = \frac{(V_{i+1}^{n+1} - V_{i-1}^{n+1}) + (V_{i+1}^n - V_{i-1}^n)}{4\Delta S}, \tag{11.2}$$

$$V = \frac{(V_i^{n+1} + V_i^n)}{2}. \tag{11.3}$$

将方程（5.1）（11.1）（11.2）和（11.3）代入 Black-Scholes 方程（1），可以得到：

$$A_i V_{i-1}^n + B_i V_i^n + C_i V_{i+1}^n = f_i, \tag{12.1}$$

其中：

$$A_i = \left(\frac{1}{4}\delta^2 i^2 - \frac{1}{4} ri\right), \tag{12.2}$$

$$B_i = \left(-\frac{1}{\Delta t} - \frac{1}{2}\delta^2 i^2 - \frac{1}{2} r\right), \tag{12.3}$$

$$C_i = \left(\frac{1}{4}\delta^2 i^2 + \frac{1}{4} ri\right), \tag{12.4}$$

$$f_i = \left(-\frac{1}{4}\delta^2 i^2 + \frac{1}{4} ri\right) V_{i-1}^{n+1} + \left(-\frac{1}{\Delta t} + \frac{1}{2}\delta^2 i^2 + \frac{1}{2} r\right) V_i^{n+1} + \left(-\frac{1}{4}\delta^2 i^2 - \frac{1}{4} ri\right) V_{i+1}^{n+1} \tag{12.5}$$

将方程（12.1）转换成矩阵形式，可以得到：

$$\begin{bmatrix} B_1 & C_1 & 0 & \cdots & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_{I-2} & B_{I-2} & C_{I-2} \\ 0 & 0 & 0 & \cdots & 0 & A_{I-1} & B_{I-1} \end{bmatrix} \begin{bmatrix} V_1^n \\ V_2^n \\ \vdots \\ V_{I-2}^n \\ V_{I-1}^n \end{bmatrix} = \begin{bmatrix} f_1^{n+1} - A_1 V_0^n \\ f_2^{n+1} \\ \vdots \\ f_{I-2}^{n+1} \\ f_{I-1}^{n+1} - C_{I-1} V_I^n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_{I-2} \\ F_{I-1} \end{bmatrix} \tag{13}$$

其中 F_i 是线性系统方程（13）的等式右侧部分，并且采用 TDMA 来求解该系统。

1.6　云并行计算实现

AzureOP 由上述三种有限差分方案的云并行计算实现组成，使用的集成开发环境是 MS Visual Studio 2011（MS-VS）。

为了在尽可能低的成本下实现快速收敛，使用 MS-MPI 库对 Black-Scholes 方程的三种有限差分方案进行并行化处理，并在微软的 Windows Azure 上运行，以获得可扩展的实例数量及满足用户对速度的需求。

在 MS-VS 上开发并行代码，首先需要安装 MPI 库，本文选择的 MPI 库是 HPC Pack 2008 R2 MS-MPI Redistributable Package。安装完成后，重复检查 MS-VS 以确保 HPC PACK 库被正确引用。然后，利用并行库 MS-VS 开发了并行显式方案、并行隐式方案和并行 Crank-Nicolson 方案。

在显式方案的方程（6.1）中，求解当前时间步长 V_i^n 的期权价格需要后续三个时间步长的值：V_{i-1}^{n+1}，V_i^{n+1} 和 V_{i+1}^{n+1}。在求解当前时间的期权价格时，由于后续时间步长的期权价格的所有值都是可用的，所以这些值可以同时求解。因此本文设计了一个如图 3 和 4 所示的云并行显式方案。

图 3 描述了显式方案的云并行算法。如图 3 所示，每个时间步长中期权价格的值是同时求解的，而不是逐一求解，并且是在当前时间步长到下一个时间步长之间的时间间隔中进行求解。

图 4 描述了 AzureOP 显式方案中的实现。如图 4 所示，MS-MPI 进程分为三类：主服务器（初始进程），网格服务器和工作节点。主服务器（初始进程）负责存储和更新期权定价的所有值，网格服务器负责计算每个网格的期权定价，节点工作人员则负责计算方程（6.2）中的 A_i，方程（6.3）中的 B_i 和方

程（6.4）中的 C_i。

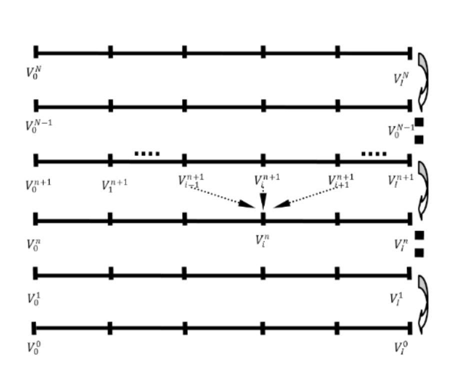


图 3　显式方案的云并行算法（算法 1）

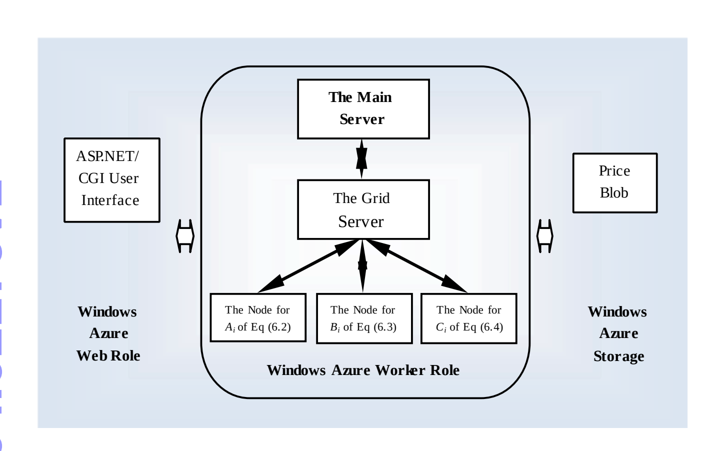


图 4　AzureOP 显式方案的实现

为了更加清晰地表达，图 3 和 4 中讨论的云并行显式方案也可以如算法 1 所述：

算法 1　基于云并行计算的显式方案。

a）发起方（主服务器）将终止条件（4）初始化为 V_i^{n+1}；

b）主服务器将方程（7.1）和（7.2）的边界条件设置为 V_i^{n+1}；

c）在每一个时间步长中：

（a）网格服务器初始化 V_i^n；

（b）计算方程（6.2）-（6.4）中的 A_i，B_i 和 C_i，随后工作节点将这些值发送给对应的网格服务器；

（c）网格服务器计算 V_i^n，并发送给主服务器；

（d）主服务器将 V_i^n 保存到 Windows Azure Storage 上的价格容器；

（e）主服务器根据 V_i^n 更新 V_i^{n+1}；

结束

在开发了算法 1 的云并行显式方案后，本文开发了云并行隐式方案。在隐式方案的线性系统方程（10）中，通过求解系统可以同时得到期权价格 V_i^n 的所有值。为了加速这一过程，在显式方案中开发的云并行实现策略的基础上，可以计算出工作节点中的系数矩阵，并解决主服务器中的线性系统问题。因此本文设计了一个如图 5 和 6 所示的云并行隐式方案。

图 5 描述了隐式方案的云并行算法。如图 5 所示，利用图 6 中列出的服务器，可以并行地计算和组装系数矩阵，当前时间步长的期权价格的所有值也可以由线性系统方程(10)求解，并且此过程一直持续。

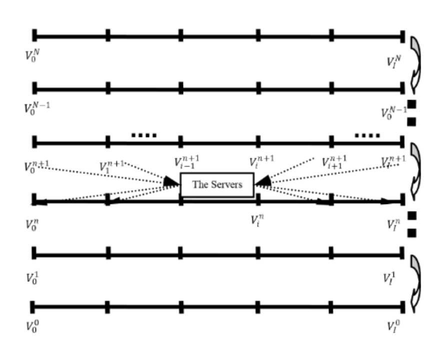


图 5　隐式方案的云并行算法（算法 2）

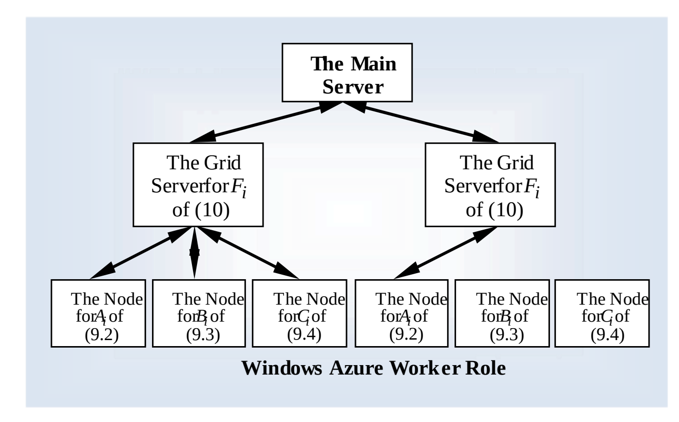


图 6　AzureOP 隐式方案的实现

图 6 描述了 AzureOP 隐式方案的实现。从图 6 可以看到，MS-MPI 进程分为三类：主服务器（发起者），网格服务器和工作节点。主服务器负责存储和更新期权价格的值，组装方程（10）的系数矩阵和其右侧矢量 F_i，并求解线性系统。网格服务器负责计算方程（10）的右侧矢量 F_i，并收集方程（9.2）的 A_i，方程（9.3）的 B_i 或方程（9.4）的 C_i。节点工作人员则负责计算方程（9.2）的 A_i，方程（9.3）的 B_i 或方程（9.4）的 C_i 的值。

云并行计算的隐式方案也可以如算法 2 所述：

算法 2　基于云并行计算的隐式方案。

a）发起方（主服务器）将终止条件（4）初始化为 V_i^{n+1}；

b）主服务器将方程（7.1）和（7.2）的边界条件设置为 V_i^{n+1}；

c）在每一个时间步长中：

（a）网格服务器初始化 V_i^n；

（b）计算方程（9.2）~（9.4）中的 A_i，B_i 和 C_i，随后工作节点将这些值发送给对应的网格服务器；

（c）网格服务器计算方程（10）中的 F_i 值，并发送给主服务器；

（d）主服务器组装系数矩阵和右侧矢量 F_i，并求解线性系统方程（10）；

（e）主服务器将 V_i^n 保存到 Windows Azure Storage 上的价格容器；

（f）主服务器根据 V_i^n 更新 V_i^{n+1}；

结束

云并行算法的 Crank-Nicolson 方案也可以类比图 5 的描述。如图 5 所示，与隐式方案类似，方程（13）中的系数矩阵被同时计算和组装，当前时间步长的所有期权价格的值由线性系统方程（13）进行求解，并且这个计算过程在所有时间步长中持

续。

图 7 描述了 AzureOP 的 Crank-Nicolson 方案的实现。主服务器负责存储和更新期权价格的值，组装系数矩阵和右侧矢量，并求解线性系统方程（13）。网格服务器负责计算右侧矢量的 F_i 值，并收集方程（12.2）的 A_i，方程（12.3）的 B_i，方程（12.4）的 C_i 或方程（12.5）的 f_i 的值。如图 7 所示，与图 6 中实现的隐式方案相同，图 7 中的工作节点负责计算方程（12.2）的 A_i，方程（12.3）的 B_i，方程（12.4）的 C_i 或方程（12.5）的 f_i 的值。

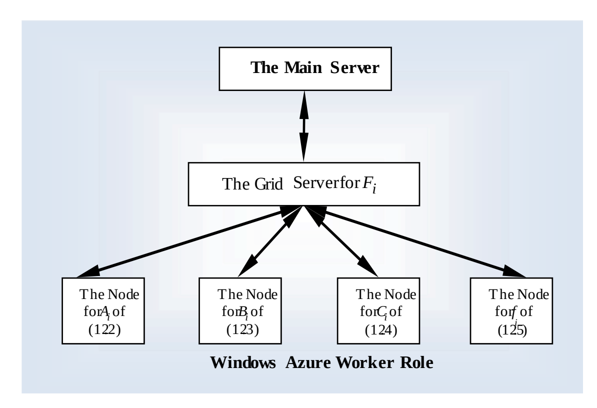


图 7　AzureOP 的 Crank-Nicolson 方案的实现

云并行算法的 Crank-Nicolson 方案也可以如算法 3 所述：

算法 3　基于云并行计算的 Crank-Nicolson 方案。

a）发起方（主服务器）将终止条件（4）初始化为 *Vin+1*；

b）主服务器将方程（7.1）和（7.2）的边界条件设置为 V_i^{n+1}；

c）在每一个时间步长中：

（a）主服务器初始化 V_i^n；

（b）计算方程（12.2）-（12.5）中的 A_i，B_i，C_i 和 f_i，随后工作节点将这些值发送给对应的网格服务器；

（c）网格服务器计算方程（13）中的 f_i 值，并发送给主服务器；

（d）主服务器组装系数矩阵和右侧矢量 f_i，并求解线性系统方程（13）；

（e）主服务器将 V_i^n 保存到 Windows Azure Storage 上的价格容器；

（f）主服务器根据 V_i^n 更新 V_i^{n+1}；

结束

1.7　Windows Azure 的安装

成功安装 AzureOP，首先需要一个 Windows Azure 帐户。之后，使用 Windows Azure HPC 调度程序来提交、管理和获取三种方案的代码结果，其中调度程序包括工作节点（头节点，计算节点和前端节点）和数据存储（Windows Azure Storage）。Windows Azure HPC 调度程序将 Windows Azure 转变为基于云的超级计算机。所有步骤都需要通过在 AppConfigure 应用程序中进行配置并填写 Windows Azure 帐户、管理员帐户以及头节点、计算节点、前端节点的数量等信息来完成的。运行命令 *hpcfwutil* 可以打开计算机节点间通信的防火墙配置。

Windows Azure HPC 调度程序准备就绪后，远程桌面连接即可访问计算节点实例。之后，开发的代码可以通过 Windows Azure Web Role 的图形用户界面提交给 Windows Azure。第一次安装之后，后续 AzureOP 的使用都不需要涉及本节中的所有步骤。

2　计算结果

本章将介绍使用 AzureOP 进行期权定价的计算结果，并将显式方案，隐式方案和 Crank-Nicolson 方案的结果一一列出。

在显式方案中，总时间为 0.01，离散化为 250、500 或 1000 步，股价 St 为 1，离散化为 25、50 或 100。AzureOP 的时间成本和费用如表 1 所示；时间步长 = 250、网格步长 = 100（I = 100）时的定价曲面计算结果如图 8 所示；显式方案在 t = 1.00，时间步长 = 250 和 I = 100 时的定价曲线如图 9 所示；显式方案在 s = 0.49（s 是股票价格的当前值），时间步长 = 250 和 I = 100 时的定价曲线如图 10 所示。

在隐式方案中，总时间为 1，离散化为 250、500 或 1000 步，股票价格 St 为 1，离散化为 25、50 或 100。AzureOP 的时间成本和费用如表 2 所示；时间步长 = 250、网格步长 = 100（I = 100）时的定价曲面计算结果如图 11 所示；隐式方案在 t = 1.00，时间步长 = 250 和 I = 100 时的定价曲线如图 12 所示；隐式方案在 s = 0.08，时间步长= 250 和 I = 100 时的定价曲线如图 13 所示。

在 Crank-Nicolson 方案中，总时间为 1，离散化为 250、500 或 1000 步，股价 St 为 1，离散化为 25、50 或 100。AzureOP 的时间成本和费用如表 3 所示；时间步长 = 250、网格步长 = 100（I = 100）时的定价曲面计算结果如图 14 所示；Crank-Nicolson 方案在 t = 1.00，时间步长 = 250 和 I = 100 时的定价曲线如图 15 所示；Crank-Nicolson 方案在 s = 0.17，时间步长 = 250 和 I = 100 时的定价曲线如图 16 所示。

本文选择的计算实例规模非常小，一共购买了 25 个实例，且共享 CPU 内核和 768 MB 内存，每小时收费 0.02 美元。

表 1 是 AzureOP 选择显式方案的计算结果。从表 1 可以得到，显式方案的时间成本从几分钟到几小时不等，而计算的开销很低。因此在保证最低费用的情况下，可以选择一个最佳的参数时间步长和网格步长，以获得良好的模拟结果，表 1 的最优选择是时间步长=1000、I = 50。

图 8 是 Azure 显式方案的计算结果。从图 8 可以看到，随着时间步长的增加，期权价格几乎保持不变，出现这种情况的原因是，为了满足显式方案稳定性的要求，本文选择了一个非常小的 Δt，约等于 10^{-5}，这导致了一个非常小的时间间隔 T = 0.01，因此显式方案几乎没有进展。

表 1　AzureOP 显式方案的时间花销和费用

时间	网格步长=25		网格步长=50		网格步长=100	
步长	时间花销/s	费用/美元	时间花销/s	费用/美元	时间花销/s	费用/美元
250	55.60	0.10	202.44	0.10	1096.21	0.10
500	219.37	0.10	791.70	0.10	4339.11	0.20
1000	877.07	0.10	3145.75	0.10	17319.28	0.50

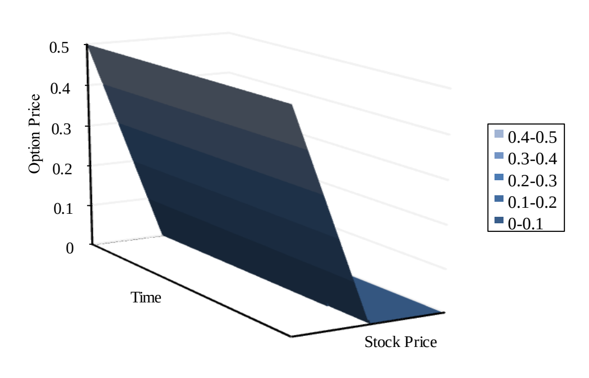


图 8　时间步长=250，网格步长=100 (T=0.01, S_t=1.00, δ=0.30, r=0.01, K=0.50)时，显式方案的定价曲面

为了比较不同时间步长的计算结果，本文绘制了 t = 1.00，以及时间步长分别为250、500和1000时的期权价格数值结果，如图 9 所示。

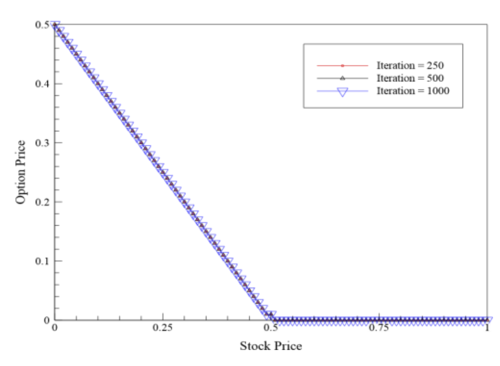


图 9　时间步长=250，网格步长=100 (t=1.00，T=0.01, S_t=1.00, δ=0.30, r=0.01, K=0.50)时，显式方案的定价曲线

图 9 描绘了显式方案的期权价格数值结果。从图 9 中可以看到，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致。时间步长的选择不会影响期权价格预测数值结果的精度，但会影响数值结果的解析度。

为了研究期权价格随时间进度产生的变化，本文绘制了 s = 0.49，以及时间步长分别为 250、500 和 1000 时的期权价格数值结果，如图 10 所示。

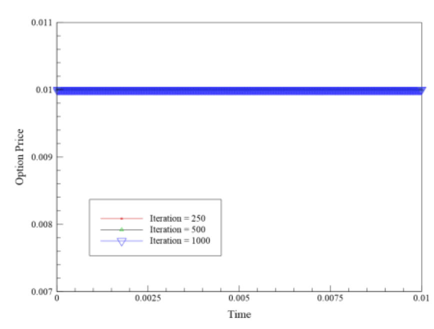


图 10　时间步长=250,股票当前价格=0.49,网格步长=100(T=0.01, St=1.00, δ=0.30, r=0.01, K=0.50)时,显式方案的定价曲线

从图 10 可以看出，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致，说明时间步长的选择不影响期权价格的数值结果。然而，随着时间的推移，期权价格几乎没有发生变化，原因正如上文讨论的。

在分析显式方案的结果之后，看一下隐式方案的结果，如表 2、图 11~13 所示。表 2 是隐式方案的计算结果。从表 2 可以看出，虽然隐式方案的时间成本从几分钟到几小时不等，但时间步长和网格步长每个组合的开销都很低。参数的最佳选择是时间步长= 1000，I = 50 和时间步长= 500，I = 100。

表 2　AzureOP 隐式方案的时间花销和费用

时间步长	网格步长=25		网格步长=50		网格步长=100	
	时间花销/s	费用/美元	时间花销/s	费用/美元	时间花销/s	费用/美元
250	50.21	0.10	168.20	0.10	655.56	0.10
500	199.07	0.10	660.41	0.10	2600.02	0.10
1000	786.28	0.10	2607.49	0.10	10282.66	0.30

比较表 1 和 2 可以得到，几乎每个时间步长和网格步长的组合中，隐式方案都比显式方案快。当期权到期时，期权价格无限趋近于 0。

图 11 是隐式方案的计算结果。从图 11 可以看出，T= 0.01 的时间间隔足够长，因此随着时间的推进，当期权到期时，期权价格接近零。比较图 8 和 11 可以得到，图 11 比图 8 更平滑，原因在于隐式方案选定的时间间隔 T 远大于显式方案的时间间隔。

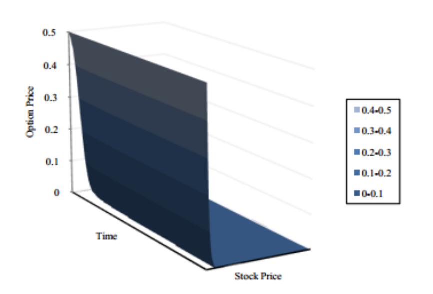


图 11　时间步长=250，网格步长=100 (T=0.01, S_t=1.00, δ=0.30, r=0.01, K=0.50)时，隐式方案的定价曲面

图 12 绘制了隐式方案的期权价格数值结果。从图 12 可以看到，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致。时间步长的选择不会影响期权价格预测数值结果的精度，但会影响数值结果的解析度。

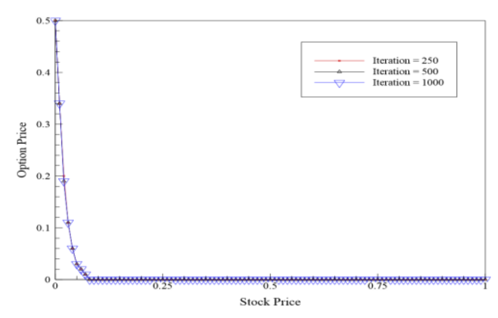


图 12　时间步长=250，网格步长=100 (t=1.00, T=1.00, S_t=1.00, δ=0.30, r=0.01, K=0.50) 时，隐式方案的定价曲线

为了研究随时间变化的特定股票价格，本文绘制了隐式方案在 s = 0.08 时的期权价格曲线，如图 13 所示。从图 13 可以

看出，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致。与显式方案相同，时间步长的选择不会影响期权价格预测数值结果的精度，但会影响数值结果的解析度。

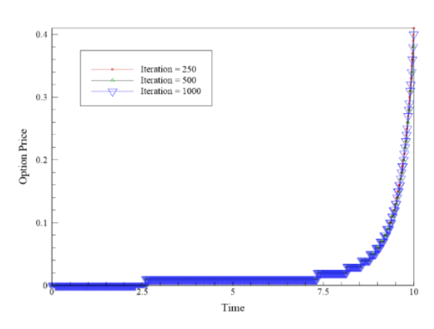


图 13 时间步长=250，股票当前价格=0.08，网格步长=100 (T=10.00, S_f=1.00, δ=0.30, r=0.01, K=0.50)时，隐式方案的定价曲线

从图 13 还可以看出，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致，这意味着时间步长的选择不会影响期权价格预测数值结果的精度，但会影响数值结果的解析度。

最后，将计算结果绘制出来，如表 3、图 14~16 所示。表 3 是 Crank-Nicolson 方案的计算结果。从表 3 可以看到，Crank-Nicolson 方案的时间成本从几分钟到几小时不等，费用也很低。时间步长和网格步长参数的最佳组合是时间步长= 1000，网格步长= 50 和时间步长= 500，网格步长= 100。

表 3 AzureOP 的 Crank-Nicolson 方案的时间花销和费用

时间	网格步长 =25		网格步长=50		网格步长=100	
步长	时间花销/s	费用/美元	时间花销/s	费用/美元	时间花销/s	费用/美元
250	57.87	0.10	198.27	0.10	782.11	0.10
500	228.35	0.10	788.86	0.10	3112.62	0.10
1000	905.65	0.10	3125.20	0.10	12418.10	0.30

图 14 是具有特定参数的 Crank-Nicolson 方案的定价曲面，从图 14 可以看到，随着时间的推移，当期权到期时，期权价格无限趋近于零。

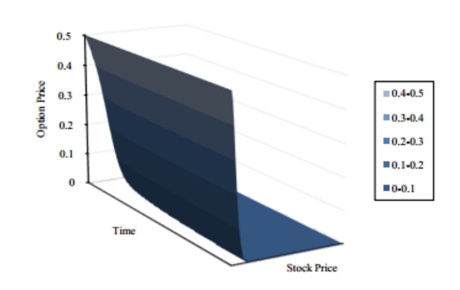


图 14 时间步长=250，网格步长=100 (T=1.00, S_f =1.00, δ=0.30, r=0.01, K=0.50)时，Crank-Nicolson 方案的定价曲面

图 15 绘制了 t = 1.00 时 Crank-Nicolson 方案的期权价格曲线。从图 15 可以看到，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致，这意味着时间步长的选择不会影响期权价格预测数值结果的精度，但会影响数值结果的解析度。

图 16 绘制了当 s = 0.17 时，Crank-Nicolson 方案的期权价格曲线。从图 16 可以看到，迭代次数等于 250、500 和 1000 的期权价格数值结果完全一致，这意味着价格曲线对时间步长并不敏感。从图 16 还可以看到，随着时间的推进，期权价格趋近于零，这与先前的假设相吻合。

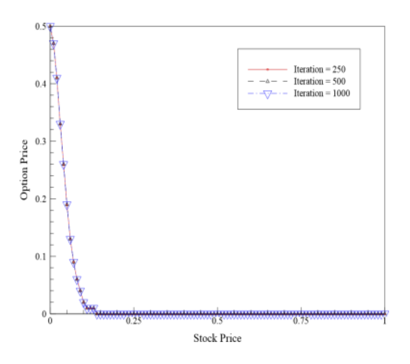


图 15 时间步长=250，网格步长=100 (t=1.00, T=1.00, S_f=1.00, δ=0.30, r=0.01, K=0.50)时，Crank-Nicolson 方案的定价曲线

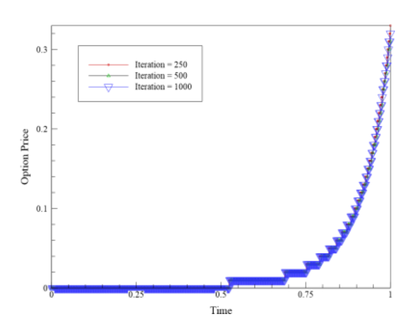


图 16 时间步长=250,股票当前价格=0.17,网格步长=100 (t=1.00, T=1.00, S_f=1.00, δ=0.30, r=0.01, K=0.50)时，Crank-Nicolson 方案的定价曲线

3 结束语

本文利用 Windows Azure 的 HPC 调度程序，开发了基于云并行计算的期权定价方案，该方案以较低的费用提供了低风险和高速度，并给出了两种方案的期权定价仿真结果。

正如上述 2.3 节中讨论的那样，主服务器（发起者）利用 TDMA 方法求解线性系统方程（10）。为了进一步加快求解线性系统的计算速度，TDMA 可以通过并行计算来实现，而现有的并行 TDMA 研究包括集群和 GPU。文献[29]中，Swarztrauber 等人开发了一个通用三对角矩阵的并行算法；文献[30]中，Nathan 等人开发了一个求解三对角矩阵的并行算法；文献[31]中，Lin 等人则开发了一个基于图的三对角矩阵算法。

期权定价的研究中，美式期权的定价问题，通常采用各种偏微分方程数值方法来求解。针对此类计算量大、计算时间长的问题，并行计算提供了有效的方法[32,33]。计算速度固然重要，但现有的美式期权定价研究主要将重心集中在数学模型的改进和提高运算速度上[34-37]，很少可以兼顾数据存储成本、计算成本和硬件开销问题。而云计算基于互联网,将规模化的计算能力、存储空间、开发平台和软件服务提供给用户,用户只需要支付一定的费用，便可以随时随地通过互联网使用云计算提供的资源

[38,39]。

金融领域云并行计算的一个关键问题是应该选择哪种部署模式：公有云还是私有云？对于没有兴趣建设和维护数据中心的金融公司来说，公有云是一个合适的选择，因为金融公司只需要向 IaaS 和 PaaS 提供商支付账单。通常来说，公有云的成本比私有云低，因为公共数据中心的规模往往比私有云大得多。此外，由于公共 PaaS 的开发人员往往来自领先的 IT 公司，因此公共 PaaS 在技术上也比私有云更先进。Windows Azure 此类云上的并行计算模式，为金融公司和专业人员提供了可扩展和低成本的并行计算，也为满足美式期权定价的速度需求提供了新的平台。而本文提供已成型的试点云软件 AzureOP，将计算和存储都进行了交付，大大提高了计算效率，缩短了执行时间，相对于自己购买各种资源来说，以较低的成本完成了复杂的、大量的计算任务，简单高效，且使用方法简便快捷。私有云则为金融公司的敏感数据提供了更多的安全保障，未来将在这一领域开展更多的项目。

参考文献：

[1] Armbrust M, Fox A, Griffith R, *et al.* A view of cloud computing [J]. Communications of the ACM, 2010, 53 (4): 50-58.

[2] Buyya R, Yeo C S, Venugopal S, *et al.* Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility [J]. Future Generation Computer Systems, 2009, 25 (6): 599-616.

[3] Marston S, Li Z, Bandyopadhyay S, *et al.* Cloud computing: the business perspective [J]. Decision Support Systems, 2011, 51 (1): 176-189.

[4] Dillon T, Wu C, Chang E. Cloud Computing: Issues and Challenges [C]// Proc of the 24th IEEE International Conference on Advanced Information Networking and Applications. 2010: 27-33.

[5] Doan D. A developer's survey on different cloud platforms [D]. San Diego: University of California, 2009.

[6] Avetisyan A I, Campbell R, Gupta I, *et al.* Open Cirrus: A Global Cloud Computing Testbed [J]. Computer, 2010, 43 (4): 35-43.

[7] Gunarathne T, Wu T L, Choi J Y, *et al.* Cloud computing paradigms for pleasingly parallel biomedical applications [J]. Concurrency & Computation Practice & Experience, 2011, 23 (17): 2338-2354.

[8] Gunarathne T, Zhang B, Wu TL, *et al.* Portable parallel programming on cloud and HPC: scientific applications of Twister4Azure [C]// Proc of the4th IEEE International Conference on Utility and Cloud Computing. Washington DC: IEEE Computer Society, 2012: 97-104.

[9] Wei L, Jackson J, Barga R. AzureBlast: a case study of developing science applications on the cloud [C]// Proc of the 1st Workshop on Scientific Cloud Computing. 2010: 413-420.

[10] Qiu X, Ekanayake J, Beason S, *et al.* Cloud technologies for bioinformatics applications [C]// Proc of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers. 2009: 1-10.

[11] Pratt B, Howbert J J, Tasman N I, *et al.* MR-Tandem: parallel X!Tandem using Hadoop MapReduce on Amazon Web Services [J]. Bioinformatics, 2012, 28 (1): 136-137.

[12] Waltz E. 1000 genomes on Amazon's cloud [J]. Nature Biotechnology, 2012, 30: 376-376.

[13] Zhang L, Gu S, Liu Y, *et al.* Gene set analysis in the cloud [J]. Bioinformatics, 2012, 28 (2): 294-295.

[14] Jourdren L, Bernard M, Dillies M A, *et al.* Eoulsan: a cloud computing-based framework facilitating high throughput sequencing analyses [J]. Bioinformatics, 2012, 28 (11): 1542-3.

[15] Langmead B, Salzberg S L. Fast gapped-read alignment with Bowtie 2 [J]. Nature Methods, 2012, 9 (4): 357-359.

[16] Hong D, Rhie A, Park S S, *et al.* FX: an RNA-Seq analysis tool on the cloud [J]. Bioinformatics, 2012, 28 (5): 721.

[17] Humphrey M, Hill Z, Ingen C V, *et al.* Assessing the value of cloudbursting: a case study of satellite image processing on Windows Azure [C]// Proc of the 7th IEEE International Conference on E-Science. 2011: 126-133.

[18] Subramanian V, Wang L, Lee EJ, *et al.* Rapid processing of synthetic seismograms using Windows Azure cloud [C]// Proc of the 2nd IEEE International Conference on Cloud Computing Technology & Science. 2011: 193-200.

[19] Rodrigues J J. Analysis of cloud-based solutions on ehrs systems in different scenarios [J]. Journal of Medical Systems, 2012, 36 (6): 3777.

[20] Lawton G. Cloud streaming brings video to mobile devices [J]. Computer, 2012, 45 (2): 14-16.

[21] Bohner M, Zheng Y. On analytical solutions of the Black–Scholes equation [J]. Applied Mathematics Letters, 2009, 22 (3): 309-313.

[22] Hull J C. Options, futures and other derivatives [M]. [S. l.] : Pearson, 2009.

[23] Gerbessiotis A V. Parallel option price valuations with the explicit finite difference method [J]. International Journal of Parallel Programming, 2010, 38 (2): 159-182.

[24] Bollerslev T, Gibson M, Zhou H. Dynamic estimation of volatility risk premia and investor risk aversion from option-implied and realized volatilities [J]. Journal of Econometrics, 2011, 160 (1): 235-245.

[25] Liu Q. Pricing American options by canonical least-squares Monte Carlo [J]. Journal of Futures Markets, 2008, 30 (2): 175-187.

[26] Jasra A, Del Moral P. Sequential Monte Carlo methods for option pricing [J]. Stochastic Analysis & Applications, 2011, 29 (2): 292-316.

[27] Devreese J P A, Lemmens D, Tempere J. Path integral approach to Asian options in the Black–Scholes model [J]. Physica A: Statistical Mechanics & its Applications, 2010, 389 (4): 780-788.

[28] Kohler M, Krzyżak A, Todorovic N. Pricing of high-dimensional american options by neural networks [J]. Mathematical Finance, 2010, 20 (3): 383-410.

[29] Swarztrauber P N. A parallel algorithm for solving general tridiagonal equations [J]. Mathematics of Computation, 1979, 33 (145): 185-199.

[30] Mattor N, Williams T J, Hewett D W. Algorithm for solving tridiagonal

matrix problems in parallel [J]. Parallel Computing, 1995, 21 (11): 1769-1782.

[31] Lin H X. A unifying graph model for designing parallel algorithms for tridiagonal systems [J]. Parallel Computing, 2001, 27 (7): 925-939.

[32] 张帆. 两类期权定价模型有限差分的并行计算 [D]. 保定：华北电力大学, 2014.

[33] 郭瑶瑶. 两类期权定价模型有限差分并行计算的新方法研究 [D]. 保定：华北电力大学, 2016.

[34] 蒋晓蓝. 用 GPU 实现基于 LSM 算法的美式期权定价 [J]. 科技创新导报, 2013, 18: 34-35.

[35] 刘颖, 江一鸣. 带不对称跳的期权定价扩散模型的参数估计 [J]. 南开大学学报, 2017, 50 (6) .

[36] 陈潇. 拟蒙特卡罗方法在期权定价中的应用 [D]. 长春：吉林大学, 2017.

[37] 代斌. 基于 GPU 的倒向随机微分方程的期权定价的并行算法研究 [D]. 济南：山东大学, 2012.

[38] 刘源, 张金燕. 云计算技术在美式期权定价中的应用 [J]. 电子技术与软件工程, 2016, 15: 205-205.

[39] 王德原. 基于期权和拍卖机制的云计算定价模型的研究 [D]. 北京：北京邮电大学, 2013.